IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received September 4, 2019, accepted September 30, 2019, date of publication October 7, 2019, date of current version October 24, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2946032

Single Machine Job Sequencing With a Restricted Common Due Window

SHIH-WEI LIN[1,2,3], KUO-CHING YING[4], WEN-JIE WU[1], AND CHEN-YANG CHENG[4]

[1]Department of Information Management, Chang Gung University, Taoyuan 333, Taiwan
[2]Department of Neurology, Linkou Chang Gung Memorial Hospital, Taoyuan 333, Taiwan
[3]Department of Industrial Engineering and Management, Ming Chi University of Technology, New Taipei 243, Taiwan
[4]Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei 106, Taiwan

Corresponding author: Kuo-Ching Ying (kcying@ntut.edu.tw)

Shih-Wei Lin and Kuo-Ching Ying are co-first authors.

The work of S.-W. Lin was supported in part by the Ministry of Science and Technology, Taiwan, under Grant MOST107-2410-H-182-005-MY2, and in part by the Linkou Chang Gung Memorial Hospital under Grant BMRPA19.

ABSTRACT This article deals with the problem of sequencing N jobs on a single machine with a restrictive common due window. The objective is to minimize the total weighted earliness-tardiness penalties, which conform to just-in-time (JIT) manufacturing. A novel backtracking simulated annealing (BSA) algorithm with a backtracking mechanism and an effective coding scheme is proposed herein to solve this problem. The performance of the proposed BSA algorithm is compared with that of the best available algorithm and the simulated annealing (SA) algorithm using four benchmark problem sets. The computational results reveal that the backtracking mechanism can improve the performance of the SA algorithm and make the proposed BSA algorithm outperform the state-of-the-art algorithm. The proposed BSA algorithm is sufficiently efficient to satisfy the real-world scheduling requirements of the JIT manufacturing system.

INDEX TERMS Scheduling, single machine, common due window, simulated annealing, backtracking.

I. INTRODUCTION

The single-machine scheduling problem (SMSP) is one of the most studied manufacturing systems owing to its practicability [1]. In recent decades, many investigations of various SMSPs have examined the dispatching rules [2], [3] and efficient heuristic algorithms [4]–[7], while applying various criteria. An increasing number of manufacturers are paying attention to just-in-time (JIT) production modes as they are confronted with numerous challenges, including increased product customization, short product life cycle and shortened time to market. In practice, meeting the specified due dates of jobs is critical for JIT production [8], [9]. Sidney [10] studied an SMSP with the objective of minimizing the total tardiness-earliness penalties for all jobs with target starting times and corresponding due dates. Gens and Levner [11] focused on minimizing the penalties of delayed jobs in an SMSP, and proposed a fast algorithm for approximating a tight bound on delay penalties. While these studies on SMSPs allowed different due dates for different jobs, Kanet [12] concentrated on a special case of the SMSP with a common due date. The goal was to minimize the mean deviation of job completion times from the common due date. Raghavachari [13] determined the common due date in an SMSP, and sequenced jobs with minimization of the weighted mean absolute deviation of job completion times. He showed that the optimal job sequence is V-shaped. A V-shaped job sequence indicates a subset of jobs that are sorted in order of non-increasing job processing times, while the remaining jobs, are sorted in order of non-decreasing job processing times. Krieger and Raghavachari [14] further revealed that the optimal schedule that minimized the sum of the penalties (early or late) for all jobs is V-shaped when all jobs have the same penalty function. Owing to its effectiveness, the V-shaped sequence has become a useful property for efficiently finding optimal or near-optimal schedules in SMSPs with variable or constant common due dates [15]–[17].

SMSPs with a common due date (CDD) have been proved to be NP-complete [18], [19]. A review of the literature by Gordon, et al. [20] included a comprehensive discussion of the computational difficulty of solving such SMSPs. In light of this difficulty of computation using exact methods (such as by the branch-and-bound algorithm [21], heuristic algorithms have become popular owing to their tractability and

The associate editor coordinating the review of this manuscript and approving it for publication was Nicola Andriolli.

efficiency, as revealed by a study on the ant colony optimization algorithm [22]. Although various SMSPs have been extensively studied, most relevant research concerns SMSPs with a CDD [23], [24]. However, the assumption of the single due date may be unrealistic in some real-world operations.

The common due window (CDW), which has various applications in JIT manufacturing, chemical processing, project scheduling, and information technology [25], is a generalization of the CDD. In this type of scheduling problem, jobs completed within a specified time interval are not penalized, while jobs completed before or after the time interval are penalized accordingly [26]. There are many use cases and fields of application regarding CDD. For example, if customers order some goods from a supplier, they generally agree to accept small deviations from a fixed delivery date due to unavoidable uncertainty such as ships waiting for embarkation in a harbor. If a bundle of goods is transported in bulk delivery, the arrival and departure times of a truck may be specified by the customers, during which time the goods must be dispatched. Furthermore, the agreement of due windows typically occurs in problems associated with tour planning [27].

In recent years, many researchers have focused on the study of SMSPs with CDWs. The most famous and interesting studies in the literature were selected and discussed as follows. Anger *et al.* [28] first introduced the concept of a common due window (CDW) to the SMSP. They revealed that the SMSP with the objective of minimizing the number of early and tardy jobs can be solved in polynomial time. Thereafter, Kramer and Lee [29] considered variable and fixed CDWs in SMSPs, and solved the related problems using a polynomial-time algorithm and a pseudo-polynomial-time algorithm, respectively. Liman and Ramaswamy [30] considered a restricted common due window (RCDW) and unrestricted common due window (UCDW) in SMSPs in which the weighted sum of earliness penalties and the weighted number of tardy jobs is minimized. A CDW problem is called restrictive if the range of the CDW influences the optimal sequence of jobs; it is called unrestrictive if the left border of the range exceeds the sum of the processing times of all jobs, or if the range of the CDW is a decision variable [27]. Liman and Ramaswamy [30] proved that the UCDW case is NP-complete and presented dynamic programming algorithms to find the optimal schedule. Ventura and Weng [31] concentrated on the RCDW in an SMSP in which the mean absolute deviation of job completion times is minimized. They presented a Lagrangian relaxation procedure and two efficient heuristics for obtaining lower bounds of the optimal solutions. Yoo and Martin-Vega [32] investigated an RCDW in SMSP with the objective of minimizing the total earliness and tardiness penalties. They showed that the problem and its similar problem with release dates can be solved in polynomial-time by a modified Moore's Algorithm [33]. Yeung *et al.* [34] further proved that the SMSP with an RCDW problem, and the objective of minimizing the total weighted earliness, tardiness and flow time penalties, becomes *NP*-hard. Biskup and Feldmann [27] investigated an RCDW in SMSP and the objective of minimizing the total weighted earliness and tardiness penalties. They supported the claim of Azizoglu and Webster [35] that an optimal solution to the SMSP with an RCDW exists in which jobs that are completed early or late exhibit the well-known V-shaped property. The orders of early and late jobs follow the dispatching rules of the weighted longest processing time (WLPT) and weighted shortest processing time (WSPT), respectively. Biskup and Feldmann [27] presented eight types of possible optimal sequences. Since the problem is NP-hard, they proposed a greedy heuristic (GH) algorithm to find initial feasible solutions, and improved the process using three meta-heuristics: the evolutionary strategy (ES), simulated annealing (SA) and threshold accepting (TA) algorithms. To demonstrate the efficiency of the GH algorithm, 250 benchmark test problems were used. However, they seem to have failed to consider that, with respect to the eight types of possible optimal sequences, the processing of the first job may straddle the boundaries of the RCDW.

On the other hand, a number of recent studies have considered the SMSP with a UCDW problem. For example, Li [36] investigated three different variants of the SMSP with a UCDW problem and batch deliveries. The objective was to minimize the total cost. He proposed polynomial-time solution procedures for the corresponding problems with significantly lower computational complexities than those of known algorithms in the literature. Liu *et al.* [37] considered the SMSP with a UCDW problem involving convex resource-dependent processing times. The objective was to minimize the total resource consumption cost under the constraint of a given schedule cost. They showed that the problem is polynomially solvable. Zhao *et al.* [38] investigated an SMSP with a UCDW, time-dependent processing times, and a controllable rate-modifying activity. The objective was to minimize the sum of earliness, tardiness, due-window-related costs and resource-related costs. They proposed a polynomial solution for the problem under consideration. Liu *et al.* [39] studied four SMSPs with a UCDW problem, where the processing time of the job was affected by the learning and positional effects. They proved that all the presented problems are polynomially solvable. Zhang *et al*. [40] studied the SMSP with a UCDW problem, linear decreasing processing times and maintenance activities, which are two common and important factors in scheduling practice. They proposed some optimality properties for the CDW assignment problem, and formulated them to obtain a polynomial time algorithm. Mor [41] extended the classical method of minmax CDD assignment and single-agent SMSPs to a setting involving two competing agents and a multi-agent setting. Furthermore, he generalized the problems to the SMSP with a UCDW problem and introduced efficient polynomial time solutions for all studied problems. Yin [42] investigated an SMSP with a UCDW and job-dependent learning effect, and showed that it can be solved in polynomial time. Wang and Li [43] dealt with four bi-criteria

TABLE 1. The computational complexity and solution algorithms of existing research.

CDW Type	Authors	Complexity	Algorithms
RCDW	Anger, et al. [28]	P	$O(n\log n)$
	Kramer and Lee [29]	P	$O(n\log n)$
	Liman and Ramaswamy [30]	P	$O(n^2 d\sum p_j)$
	Biskup and Feldmann [27]	NP-hard	Heuristic
	Ventura and Weng [31]	NP-hard	Heuristic
	Yoo and Martin-Vega [32]	P	Heuristic
	Yeung et al. [34]	P	Dynamic Programming
	Azizoglu and Webster [35]	NP-hard	Branch-and-Bound
UCDW	Mor [26]	P	$O(n)$, $O(n^3)$
	Liman and Ramaswamy [30]	NP-hard	$O(n^2 D\sum p_j)$
	Li [36]	P	$O(n^4)$, $O(n^5)$
	Liu et al. [37]	P	$O(n\log n)$
	Zhao et al. [38]	P	$O(n^4)$
	Liu et al. [39]	P	$O(n^3)$, $O(n\log n)$
	Zhang et al. [40]	P	$O(n\log n)$
	Mor [41]	P	$O(\max\{n^A, n^B\})$, $O((m^A+m^B)n^K+(m^B\log m^B)+(m^A)^2)$
	Yin [42]	P	$O(n^3)$
	Wang and Li [43]	P, NP-hard	$O(n)$, Mixed Integer Linear Programming
	Wang et al. [44]	P	$O(n\log n)$, $O(n^3)$

SMSPs with a UCDW problem and resource-dependent processing times, in which the resource amounts assigned to the jobs can be either discrete or continuous. The authors proposed pseudo-polynomial-time algorithms and an optimal algorithm, which can help practitioners addressing corresponding problems faced in their specific environments. At the same year, Mor [26] studied two extensions of min-max SMSP with a UCDW problem. The first problem is to minimize the maximum scheduling cost subject to maximal resource consumption; the second one is to minimize the resource consumption subject to an upper bound on the scheduling measure. It was proved that both considered problems are polynomially solvable. Wang *et al.* [44] dealt with an SMSP with a UCDW problem, in which the objective was to minimize the total position-dependent weighted cost. A polynomial time solution algorithm was provided for the corresponding problem.

The computational complexity and solution algorithms of existing research for SMSPs with RCDW and UCDW are summarized in Table 1. Generally, it can be seen in Table 1 that there are many pseudo-polynomial-time algorithms for the polynomially solvable problems, but only a few meta-heuristic algorithms for the NP-hard problems. For further detailed discussion on SMSCDWAPs, the reader is referred to the recent survey article by Janiak *et al.* [25].

Motivated by the excellent research of Biskup and Feldmann [27], this study focuses on the SMSP with an RCDW in which the total weighted earliness-tardiness penalties are minimized. A novel backtracking simulated annealing (BSA) algorithm, which uses a backtracking mechanism to escape from local optima sequences and an effective coding scheme to find possible optimal sequences and waiting times, is proposed herein. Twelve types of possible optimal sequences are presented in calculating the total weighted earliness-tardiness penalties. The performance of the proposed BSA algorithm is demonstrated by comparing its computational results with those obtained using the state-of-the-art ES algorithm [27] and the simulated annealing (SA) algorithm in solving four sets of benchmark problems. The remainder of this paper is organized as follows. The following section defines the considered SMSP. Section 3 discusses 12 types of possible optimal sequences and the formulae for the corresponding objective functions. Section 4 describes in detail the proposed BSA algorithm. Section 5 presents the computational experiments and results obtained using four benchmark problem sets. Finally, Section 6 draws conclusions and offers suggestions regarding directions for future research.

II. PROBLEM STATEMENT, DEFINITIONS AND NOTATION

The SMSP with an RCDW in this work is described as follows. The following notations are used.

Notations:

- j job index
- α_j unit penalty associated with the earliness of job J_j
- β_j unit penalty associated with the tardiness of job J_j
- p_j processing time of job J_j
- C_j completion time of job J_j

d_E earliest due date
d_T latest due date
C_E earliest possible completion time of all jobs
h_E given parameters that determine d_E
h_T given parameters that determine d_T
E_j earliness of job J_j
T_j tardiness of job J_j

Consider a set of N jobs $\mathbf{J} = \{J_j | j = 1, \ldots, N\}$ to be processed on a single machine with an RCDW. The objective is to determine the sequence of all jobs that minimizes the total weighted earliness-tardiness penalties. By applying the three-field classification scheme of Graham *et al.* [45], the addressed SMSP can be expressed as the triplet $1|RCDW|\sum(\alpha_j E_j + \beta_j T_j)$, where E_j and T_j are the earliness and tardiness of job J_j ($j = 1, 2, \ldots, N$), respectively, and α_j and β_j are the unit penalties (penalty weights) associated with the earliness and tardiness of job J_j, respectively. Let $p_j, j = 1, \ldots, N$, be the processing time of job J_j, and C_j, $j = 1, \ldots, N$, be the completion time of job J_j. For the RCDW, let d_E and d_T represent the earliest (left boundary) and latest (right boundary) due dates, respectively. With reference to Feldmann and Biskup [46], the size and position of the RCDW, based on d_E and d_T, are predetermined as:

$$d_E = \lfloor h_E \cdot C_E \rfloor = \left\lfloor h_E \cdot \sum_{j=1}^{N} p_j \right\rfloor \tag{1}$$

$$d_T = \lfloor h_T \cdot C_E \rfloor = \left\lfloor h_T \cdot \sum_{j=1}^{N} p_j \right\rfloor \tag{2}$$

where C_E is the earliest possible completion time of all jobs, and h_E and h_T are the given parameters that determine d_E and d_T, respectively.

Throughout the paper, parameters h_E and h_T satisfy the inequality $0 < h_E < h_T < 1$ such that $(d_T - d_E) < C_E$. Additionally, the latest due date satisfies $d_T \geq \min_{j=1,\ldots,N} p_j$; otherwise, an optimal sequence can be easily obtained by sequencing all jobs in order of non-decreasing p_j / β_j. The critical assumptions made in the $1|RCDW|\sum(\alpha_j E_j + \beta_j T_j)$ problem herein are described as follows.

- All jobs are independent of each other and processed consecutively on one machine.
- The first job in a production sequence may be processed after the beginning of the scheduling horizon, which is at time zero.
- The machine can only process a job once and must process all jobs without any interruption from the beginning of the processing of the first job to the completion of the last job.
- The setup time of the machine is negligible.
- No job is interrupted and no machine breaks down.
- The size and position of the RCDW are predetermined and fixed.
- The RCDW is smaller than the makespan of the N jobs.
- The latest due date (right boundary) of the RCDW is after the earliest possible completion of any one job.

III. TWELVE TYPES OF POSSIBLE OPTIMAL SEQUENCES

With respect to the $1|RCDW|\sum(\alpha_j E_j + \beta_j T_j)$ problem, Biskup and Feldmann [27] discussed eight types of possible optimal sequences. The orders of early and tardy jobs follow the WLPT and WSPT rules, respectively (and so exhibit V-shaped property). For ease of explanation, let S_j ($j = 1, 2, \ldots, N$) be the starting time of job J_j; $\mathbf{E} = \{J_j | C_j \leq d_E\}$, $\mathbf{W} = \{J_j | S_j \geq d_E,\ C_j \leq d_T\}$, and $\mathbf{T} = \{J_j | S_j \geq d_T\}$ denote the sets of non-straddling jobs with starting and completion times before, within and after the RCDW, respectively. Then, an optimal sequence exhibits the following well-known properties [35].

Property 1: In an optimal sequence, jobs must be in a V-shaped arrangement, meaning that jobs in set $\mathbf{E}$ (or $\mathbf{T}$) are ordered by non-increasing (or non-decreasing) ratio p_j / α_j (or p_j / β_j).

Property 2: An optimal sequence exists in which either the job in the first position begins at time zero or one job is completed at d_E or d_T.

Property 1 implies that one or two straddling jobs may be present in the optimal sequence, and Property 2 means that an optimal sequence may exist in which all jobs have production waiting times. Given these two properties, twelve types of possible optimal sequences are provided, presented in Fig. 1. Therein, J_E (with $S_j < d_E \vee d_E < C_j \leq d_T$), J_T (with $d_E \leq S_j < d_T \vee C_j > d_T$) and J_B (with $S_j < d_E \vee C_j > d_T$) represent left-straddling, right-straddling and double-straddling individual jobs, with starting and completion times that straddle the RCDW boundaries d_E, d_T and both d_E and d_T, respectively. Note that the twelve cases are established under the following assumptions:

- $d_E \geq \min\{p_1, p_2, \ldots, p_N\}$; otherwise, the problem is trivial. The optimal solution is ordering the jobs according to non-decreasing ratios p_j / β_j and starting the first job at time zero.
- $d_T - d_E < \sum_{j=1,\ldots,N} p_j$; otherwise, the problem becomes trivial.

Case 1 ($\mathbf{W} = \phi$) involves production waiting time and right-straddling job J_T. Case 2 ($\mathbf{E} = \mathbf{W} = \phi$) involves left-straddling job J_E and right-straddling job J_T. Case 3 ($\mathbf{E} = \mathbf{W} = \phi$) involves double-straddling job J_B. Cases 4 and 5 involve production waiting times, and Case 4 involves left-straddling job J_E. In Case 6, all jobs are non-straddling jobs. Case 7 involves right-straddling job J_T with production waiting time. In Cases 8 to 12, production begins at time zero; Cases 8 and 9 involve the left-straddling job J_E and right-straddling job J_T, respectively. Cases 10 and 11 ($\mathbf{E} = \phi$) involve both a left-straddling job J_E and a right-straddling job J_T. Case 12 ($\mathbf{W} = \phi$) involves the double-straddling job J_B. In more detail, the first, seventh and ninth cases are characterized by the existence of one job completed exactly in d_E; the fourth and eighth cases are characterized by the

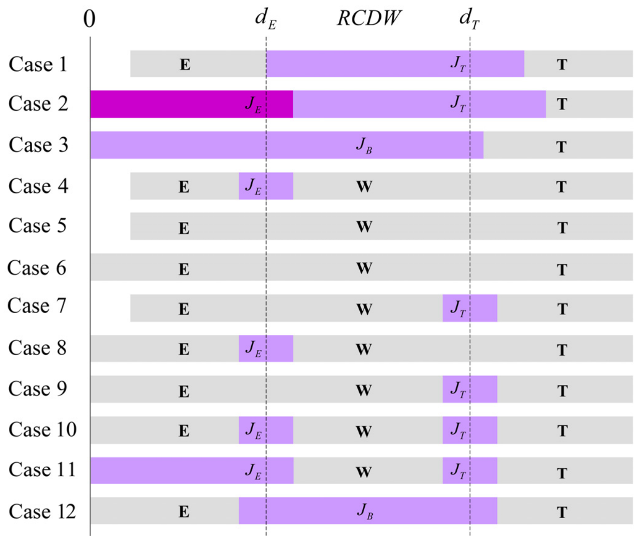

FIGURE 1. Twelve types of possible optimal sequences.

existence of one job completed exactly in d_T; the fifth and sixth cases are characterized by the existence of one job completed exactly in d_E and another job completed exactly in d_T. In the remaining five cases: cases 2, 3, 10, 11 and 12, at least one straddling job occurs. In the third and twelfth cases a double-straddling job occurs. These straddling jobs are stressed by shading. In cases 1, 3, 4, 7, 8, 9 and 12 only one straddling job emerges and in cases 2, 10 and 11 two straddling jobs occur. Moreover, in the first, fourth, fifth, and seventh cases, their first job of set **E** starts later than time point zero. To simulate the leading idle time of these cases, set **E** is moved slightly to the right in Fig. 1. It is noted that for an optimal solution the existence of two straddling jobs or a double-straddling job are inconsistent with leading idle time (Property 2). Otherwise, the total weighted earliness-tardiness penalties could be decreased by moving all jobs to the left or to the right. Additionally, as seen in cases 1, 2, 3, 10, 11 and 12, sets **E** and **W** can be empty, but we have to mention that set **T** cannot be empty, as an empty set **T** contravenes the assumption that the CDW is restrictive.

These twelve types provide a more complete and accurate perspective on all possible optimal sequences associated with various straddling jobs and production waiting times. To facilitate the proposed BSA algorithm to evaluate possible candidate solutions, these cases are classified into six groups (***G1-G6***); each is associated with a formula for the total weighted earliness-tardiness penalties, as follows.

G1. Case 1: $\sum_{j\in\mathbf{E}}\alpha_j(d_E-C_j)+\beta(J_T)\left[p(J_T)+d_E-d_T\right]+\sum_{j\in\mathbf{T}}\beta_j\left[p(J_T)+d_E-d_T+p_j\right]$

G2. Case 2: $\beta(J_T)\left[p(J_E)+p(J_T)-d_T\right]+.\sum_{j\in\mathbf{T}}\beta_j\left[p(J_E)+p(J_T)-d_T+p_j\right]$

G3. Case 3: $\beta(J_B)\left[p(J_B)-d_T\right]+\sum_{j\in\mathbf{T}}\beta_j\left[p(J_B)-d_T+p_j\right]$.

G4. Cases 4 to 6: $\sum_{j\in\mathbf{E}}\alpha_j(d_E-C_j)+\sum_{j\in\mathbf{T}}\beta_j p_j$.

G5. Cases 7 to 11: $\sum_{j\in\mathbf{E}}\alpha_j(d_E-C_j)+\beta(J_T)\left[p(J_T)+d_E-d_T\right]+\sum_{j\in\mathbf{T}}\beta_j\left[p(J_T)+d_E-d_T+p_j\right]$

G6. Case 12: $\sum_{j\in\mathbf{E}}\alpha_j(d_E-C_j)+\beta(J_B)T(J_B)+\sum_{j\in\mathbf{T}}\beta_j\left[T(J_B)+p_j\right]$
where $T(J_B)=\sum_{j\in\mathbf{E}}p_j+p(J_B)-d_T$.

IV. PROPOSED BSA ALGORITHM

This work develops a novel SA-based heuristic, called backtracking simulated annealing (BSA), to solve the $1|RCDW|\sum(\alpha_j E_j+\beta_j T_j)$ problem. The SA algorithm is a well-known local search-based meta-heuristic that can escape from the local optima by accepting, with small probability, worse solutions during the search process. This famous algorithm has been successfully used to solve many hard combinatorial optimization problems, such as neural net [47], benchmark functions [48], image restoration problem [49], 0-1 Knapsack Problem [50], and quadratic assignment problem [51]. An SA algorithm typically begins with a randomly generated initial solution. Then, at each iteration, it finds a solution in the neighborhood of the current solution. If the new solution is better than the current solution, it replaces the latter with the former and the search process resumes from the new current solution. It also allows a worse neighborhood solution to replace the current solution, with a small probability, so that the procedure can escape local optima at which it may otherwise become trapped. The proposed BSA algorithm applies a backtracking mechanism to escape from the local optima sequences and an effective coding scheme to search for possible optimal sequences and the waiting time for the $1|RCDW|\sum(\alpha_j E_j+\beta_j T_j)$.

The following subsections describe the solution representation and coding procedures, the neighborhood solutions, the parameters used in the proposed BSA algorithm, and the procedure of its implementation.

A. SOLUTION REPRESENTATION AND CODING PROCEDURE

In this study, a solution is coded using a non-negative integer value to represent the waiting time, LT ($0\le LT\le d_E$), and n integers to specify an ordered list of n jobs. Given a waiting time and an ordered list, the corresponding solution Π is coded using the following two steps. In the first step, the completion time $C_{[j]}$ ($j=1,\ldots,N$) of the j^{th} job in an ordered list is calculated as follows:

$$C_{[1]}=LT+p_{[1]} \tag{3}$$

$$C_{[j+1]}=C_{[j]}+p_{[j+1]} \tag{4}$$

Based on its completion time, each job is identified as being a member of one of the three sets of non-straddling job (**E**, **W**, and **T**) or one of the three straddling jobs (J_E, J_T and J_B). In the second step, the jobs in **E** are re-arranged in order of non-increasing ratio p_j/α_j, while the jobs in **T** are re-arranged in order of non-decreasing ratio p_j/β_j. Since the value of the objective function must be computed frequently in the search process, the method for quickly sorting

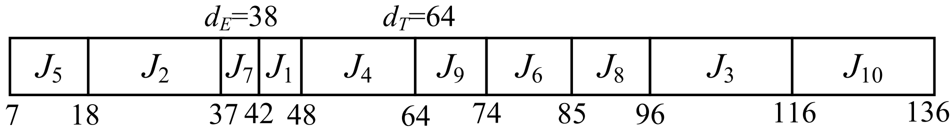

FIGURE 2. The Gantt chart of the optimal solution.

Begin
Input T_0, T_f, I_{iter}, α, $\eta_{\min}$, $B_{non\text{-}improving}$ and problem instance;
Generate initial solution Π by RGH heuristic
$T \leftarrow T_0$; $B_{non} \leftarrow 0$; $\Pi_{Best} \leftarrow \Pi$; $\eta_r = \eta_{\min}$ (r=1, 2, 3)
while ($T > T_f$) **do**
  for (i=0; $i \le I_{iter}$; i++)
    Generate $\rho \leftarrow random()$;
    Let LT be the leading idle time of solution Π;
    $\eta_T = \eta_1 + \eta_2 + \eta_3$;
    if ($\rho < \eta_1 / \eta_T$ and $LT < d_E$) { $LT \leftarrow LT$+1; $r\_index \leftarrow 1$;}
    else if ($\rho > (\eta_1 + \eta_2)/\eta_T$ and LT>0) { $LT \leftarrow LT$-1; $r\_index \leftarrow 2$;}
    else { $r\_index \leftarrow 3$;}
   Randomly choose two jobs in different set (E, W, T) using new LT and swap them to obtain new solution Π_{new};
   $\Delta E \leftarrow obj(\Pi_{new}) - obj(\Pi)$;
   if ($\Delta < 0$) **then** $\eta_{r\_index} \leftarrow \eta_{r\_index} + 1$;
   else if ($\Delta > 0$) **then** $\eta_{r\_index} \leftarrow \eta_{r\_index} - 1$;
   else $\eta_{r\_index} \leftarrow \eta_{r\_index} - 0.1$;
   if ($\eta_{r\_index} < \eta_{\min}$) **then** { $\eta_{r\_index} \leftarrow \eta_{\min}$;}
   if ($\Delta \le 0$) **then** $\Pi = \Pi_{new}$;
   else {
     Generate $\gamma \leftarrow random()$;
     if ($\gamma < e^{-\Delta E/T}$) **then** { $\Pi \leftarrow \Pi_{new}$;}
   }
   if ($obj(\Pi) < obj(\Pi_{Best})$) **then** { $\Pi \leftarrow \Pi_{Best}$; $B_{non} \leftarrow 0$ };
   else { $B_{non} \leftarrow B_{non}$+1;}
   if ($B_{non} = I_{iter} \times B_{non-improving}$) **then** { $B_{non} \leftarrow 0$; $\Pi \leftarrow \Pi_{Best}$;}
  end for
  $T \leftarrow \alpha T$;
end while
End

FIGURE 3. Pseudo-code of the proposed BSA algorithm.

jobs in a V-shape that was proposed by Lin, et al. [52] is used. This method is performed using two pre-established lookup tables to quickly determine the sequences of jobs in **E** and **T**. Procedures and an example of its use were presented by Lin, et al. [52]. After the jobs in **E** and **T** are sorted in a V-shape, the ordered list in the solution is coded as ($\mathbf{E}, J_E, \mathbf{W}$(or J_B), $J_T, \mathbf{T}$). Notably, if J_B exists, then $\mathbf{W} = \phi$.

The coding procedure is demonstrated by applying it to a random generated instance (see Table 2) with ten-job, $d_E = 38$ and $d_T = 64$. Given a waiting time of seven and a permutation list (5, 2, 7, 1, 4, 3, 9, 10, 6, 8), from Eqs. (3) and (4) , $C_5 = 7 + 11 = 18$, $C_2 = 18 + 19 = 37$, $C_7 = 37 + 5 = 42$, $C_1 = 42 + 6 = 48$, $C_4 = 48 + 16 = 64$, $C_3 = 64 + 20 = 84$, $C_9 = 84 + 10 = 94$, $C_{10} = 94 + 20 = 114$, $C_6 = 114 + 11 = 125$, $C_8 = 125 + 11 = 136$. Therefore, $\mathbf{E} = \{J_5, J_2\}$, $J_E = J_7$, $J_B = \Phi$, $J_T = \Phi$, $\mathbf{W} = \{J_1, J_4\}$, $\mathbf{T} = \{J_3, J_9, J_{10}, J_6, J_8\}$, and the right straddling or double-straddling job does not exist. Finally, by sorting the jobs in **E** and **T** in a V-shape, the ordered list in the solution

TABLE 2. Data for a random generated instance.

	1	2	3	4	5	6	7	8	9	10
p_j	6	19	20	16	11	11	5	11	10	20
α_j	5	8	5	8	3	6	9	7	10	5
β_j	9	12	1	15	12	1	13	1	2	1

is recoded as (5, 2, 7, 1, 4, 9, 6, 8, 3, 10), and the solution is then coded as $\Pi = (7|\ 5, 2, 7, 1, 4, 9, 6, 8, 3, 10)$. The Gantt chart of this solution is shown in Fig. 2. This solution is indeed the optimal solution of the above problem instance.

B. NEIGHBORHOOD

The change operator of the *LT* and the job swap operator are used to generate the solution from the neighborhood of the current solution Π. The set of solutions in the neighborhood of the current solution Π is denoted as $\mathcal{N}(\Pi)$. In each iteration, the *LT* is determined by applying one of three rules: increase one (R_1), reduce one (R_2), and keep the same (R_3) the current *LT*, according to the formula $prob_r = \eta_r / \sum_{l=1}^{3} \eta_l, (r = 1, 2, 3)$, where $prob_r$ is the probability of choosing rule R_r and η_r is the fitness value of rule R_r, which is auto-tuned in each iteration according to the following criteria:

(1) If the current solution is improved and updated to a new obtained solution that is generated by applying a selected rule (R_r), then set $\eta_r =: \eta_r + 1$;
(2) Otherwise, if the new obtained solution that is generated by applying a selected rule (R_r) is worse than the current solution, then set $\eta_r =: \eta_r - 1$. If $\eta_r < \eta_{\min}$, and then $\eta_r =: \eta_{\min}$, where $\eta_{\min}$ is the minimal allowed value of $\eta_r (r = 1, 2, 3)$.
(3) Otherwise, $\eta_r =: \eta_r - 0.1$.

Notably, in the application of the three *LT* updating rules, the possible range of *LT* ($0 \le LT \le d_E$) must be considered. If the *LT* is zero or greater than d_E, the R_2 and R_1 cannot be applied, respectively. For example, if current *LT* is 7, new *LT* will be 8, 6, and 7 for R_1, R_2, and R_3 rule, respectively. After the *LT* is changed, a new feasible solution Π_{new} is generated from $\mathcal{N}(\Pi)$ by randomly choosing and swapping the i^{th} and the j^{th} positions of jobs in Π. Notably, if the selected jobs are in the same set **E**, **W**, or **T**, the sequence of jobs in Π cannot be improved; therefore, two jobs may not be selected from a single set. For example, if $\Pi = (7|5, 2, 7, 1, 4, 9, 6, 8, 3, 10)$, then $\mathbf{E} = \{J_5, J_2\}, J_E = J_7, J_B = \mathbf{\Phi}, J_T = \mathbf{\Phi}$, $\mathbf{W} = \{J_1, J_4\}, \mathbf{T} = \{J_3, J_9, J_{10}, J_6, J_8\}$. Swap J_9 and J_{10} will not change the objective function value of Π because the V-shape property is applied; therefore, two jobs which are not in the set can be swapped.

C. BSA PROCEDURES

The pseudo-code of the proposed BSA algorithm is shown in Fig. 3. Let T_0 and T_f represent the initial and final temperatures, respectively; I_{iter} denotes the total number of iterations that the perturbation should repeat at a certain temperature; α indicates the control coefficient of the cooling schedule; $\eta_{\min}$ represents the minimal value of $\eta_r (r = 1, 2, 3)$, where η_r is the fitness value of choosing rule R_r; $B_{non-improving}$ stands for the cumulative number of consecutive temperature reductions. If the best value of the objective function is not improved by $B_{non-improving}$ consecutive temperature reductions, then the incumbent solution will be backtracked to the current best solution. The detailed procedures of the proposed BSA algorithm to be used to solve the $1|RCDW|\sum(\alpha_j E_j + \beta_j T_j)$ problem are described as follows.

Initially, the current temperature T is set to T_0 and an initial solution Π is obtained using the revised greedy heuristic (RGH) [53]. The value of the objective function of Π is denoted as $obj(\Pi)$. The current best solution Π_{best} is set to Π, and $obj(\Pi_{best})$ is initialized as $obj(\Pi)$.

At each iteration, a neighborhood solution Π_{new} with waiting time is generated from $\mathcal{N}(\Pi)$, and its objective function value is evaluated. If $obj(\Pi_{new})$ is not worse than $obj(\Pi)$, then Π_{new} replaces Π as the incumbent solution. Otherwise, Π_{new} is accepted as the incumbent solution with a small probability. This probability is typically calculated using the Boltzmann function. More specifically, let $\Delta E = obj(\Pi_{new}) - obj(\Pi)$; then the probability of replacing Π with a worse neighborhood solution Π_{new} is $e^{(-\Delta E/T)}$. Such a replacement is implemented by randomly generating a number $0 < r < 1$ and replacing Π with Π_{new} when $r < e^{(-\Delta E/T)}$.

The current temperature T decreases after I_{iter} iterations at the current temperature, according to the formula $T =: \alpha T$, $0 < \alpha < 1$. If Π_{best} is not improved in $B_{non-improving}$, then the backtracking mechanism is implemented by setting the current solution Π to Π_{best}. The searching procedure terminates when the current temperature is lower than the final temperature T_F, and the best solution is then output.

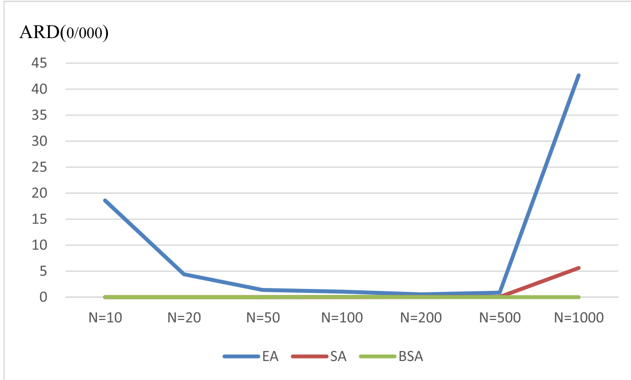

FIGURE 4. The average ARDs of all compared algorithms.

V. COMPUTATIONAL EXPERIMENTS AND RESULTS

The performance of the proposed BSA algorithm is compared with those of SA (BSA without a backtracking mechanism) and EA [27], which is the best available algorithm published in the literature. All of the BSA, SA, and EA algorithms

TABLE 3. ARDs of the compared algorithms for each size of test problem.

Problem Set	N	ES		SA		BSA	
		ARD	Time*	ARD	Time	ARD	Time
I	10	3.019	0.010	0.000	0.046	0.000	0.101
	20	0.000	0.047	0.000	0.140	0.000	0.185
	50	1.230	0.435	0.000	0.622	0.000	0.662
	100	0.415	2.219	0.010	2.031	0.053	2.065
	200	0.233	13.328	0.012	7.398	0.029	7.457
	Average	0.979	-	0.004	-	0.016	-
II	500	1.459	173.8	0.034	43.6	0.003	44.0
	1000	74.079	1302.5	0.021	169.1	0.001	169.4
	Average	37.769	-	0.027	-	0.002	-
III	10	34.165	0.011	0.000	0.045	0.000	0.100
	20	8.764	0.053	0.000	0.140	0.000	0.186
	50	1.513	0.455	0.000	0.641	0.000	0.686
	100	1.689	2.404	0.000	2.156	0.000	2.205
	200	0.842	14.276	0.011	8.009	0.009	8.012
	Average	34.165	-	0.000	-	0.000	-
IV	500	0.292	186.9	0.037	46.6	0.003	47.1
	1000	11.197	1422.3	11.197	180.3	0.000	181.3
	Average	5.744	-	5.617	-	0.002	-
Total Average		9.921	-	0.809	-	0.007	-

*: CPU time in seconds.

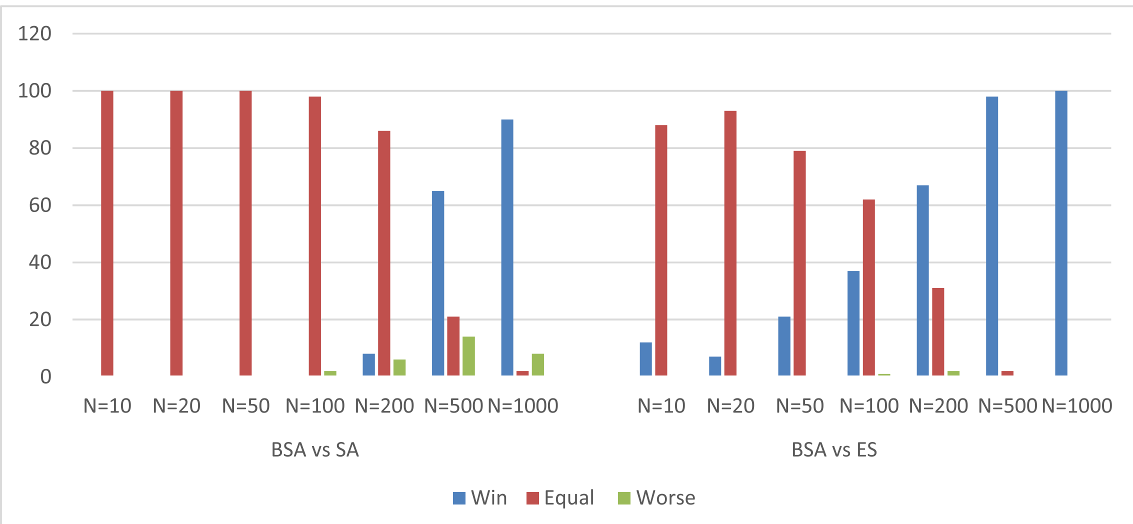

FIGURE 5. The number of solutions obtained by BSA is better, equal to, or worse than those obtained by SA and EA.

utilize an initial solution that is obtained using RGH [53]. The proposed BSA algorithm was coded using C language and executed on a personal computer with an Intel Core 2 i7-920 2.67 GHz CPU and 4 GB of RAM. EA was re-coded and run on the same computer; the computational times were then compared. The following subsection describes the

TABLE 4. Paired *t*-tests on ARDs of the compared algorithms for each size of test problem.

BSA *vs.* SA	Problem Set	N	Mean	S-Dev	t-value	DF	P-value	Significant
	I	10	0.000	0.000	-	49	-	No
		20	0.000	0.000	-	49	-	No
		50	0.000	0.000	-	49	-	No
		100	-0.043	0.215	1.764	49	0.08397	No
		200	-0.017	0.122	1.432	49	0.15845	No
	II	500	0.031	0.062	3.761	49	0.00045	Yes
		1000	0.020	0.024	6.070	49	0.00000	Yes
	III	10	0.000	0.000	-	49	-	No
		20	0.000	0.000	-	49	-	No
		50	0.000	0.000	-	49	-	No
		100	0.000	0.000	-	49	-	No
		200	0.002	0.038	0.956	49	0.34399	No
	IV	500	0.033	0.056	4.468	49	0.00005	Yes
		1000	11.197	32.536	2.707	49	0.00932	Yes
BSA *vs.* ES	**Problem Set**	**N**	**Mean**	**S-Dev**	**t-value**	**DF**	**P-value**	**Significant**
	I	10	3.019	0.000	1.423	49	0.16111	No
		20	0.000	0.000	-	49	-	No
		50	1.230	0.000	2.624	49	0.01155	Yes
		100	0.355	0.233	2.786	49	0.00758	Yes
		200	0.204	0.086	3.949	49	0.00025	Yes
	II	500	1.456	2.052	5.216	49	0.00000	Yes
		1000	74.078	35.976	14.816	49	0.00000	Yes
	III	10	34.165	0.000	3.451	49	0.00116	Yes
		20	0.000	0.000	2.537	49	0.01443	Yes
		50	1.513	0.000	2.954	49	0.00481	Yes
		100	1.656	0.011	3.934	49	0.00026	Yes
		200	0.833	0.038	6.396	49	0.00000	Yes
	IV	500	0.288	0.285	7.357	49	0.00000	Yes
		1000	11.197	32.536	2.645	49	0.01096	Yes

benchmark problems, parameter value determination, and computational results.

A. BENCHMARK PROBLEMS

To evaluate the performance of the proposed BSA algorithm in solving the $1|RCDW|\sum(\alpha_j E_j + \beta_j T_j)$ problem, four problem sets (I, II, III, and IV) extended from the well-known benchmark problems [54] are used. The Problem set I comprises the benchmark problems with the number of jobs $N = \{10, 20, 50, 100, 200\}$, and possible combinations of RCDW parameters $(h_E, h_T) = \{(0.1, 0.2), (0.1, 0.3), (0.2, 0.5), (0.3, 0.4), (0.3, 0.5)\}$. Ten benchmark problems are generated for each combination yielding a total of 250 benchmark problems in the problem set I. Notably, problem set I was also used in tests by Biskup and Feldmann [53] and Ying *et al.* [54]. The experimental design of problem set II is the same as that of problem set I, but with $N = \{500, 1000\}$. Therefore, problem set II comprised 100 benchmark problems. The experimental designs of problem sets III and IV are the same as those of problem sets I and II, respectively, except that $(h_E, h_T) = \{(0.4, 0.5), (0.4, 0.6), (0.5, 0.6), (0.5, 0.7), (0.6, 0.7)\}$. As a result, a total of 700 benchmark problems are considered.

The performance of the compared algorithms is evaluated using the average relative deviation (ARD), defined as

TABLE 5. The number and percentage of solutions obtained by the BSA are better, equal to, or worse than those obtained by SA and ES algorithms.

Problem Set	N	BSA *vs.* SA						BSA *vs.* ES					
		Better		Equal		Worse		Better		Equal		Worse	
I	10	0	0.00%	50	100.00%	0	0.00%	1	2.00%	49	98.00%	0	0.00%
	20	0	0.00%	50	100.00%	0	0.00%	0	0.00%	50	100.00%	0	0.00%
	50	0	0.00%	50	100.00%	0	0.00%	9	18.00%	41	82.00%	0	0.00%
	100	0	0.00%	48	96.00%	2	4.00%	14	28.00%	35	70.00%	1	2.00%
	200	4	8.00%	42	84.00%	4	8.00%	28	56.00%	21	42.00%	1	2.00%
	Sub total	4	1.60%	240	96.00%	6	2.40%	52	20.80%	196	78.40%	2	0.80%
II	500	29	58.00%	13	26.00%	8	16.00%	50	100.00%	0	0.00%	0	0.00%
	1000	40	80.00%	2	4.00%	8	16.00%	50	100.00%	0	0.00%	0	0.00%
	Sub total	69	69.00%	15	15.00%	16	16.00%	100	100.00%	0	0.00%	0	0.00%
III	10	0	0.00%	50	100.00%	0	0.00%	11	22.00%	39	78.00%	0	0.00%
	20	0	0.00%	50	100.00%	0	0.00%	7	14.00%	43	86.00%	0	0.00%
	50	0	0.00%	50	100.00%	0	0.00%	12	24.00%	38	76.00%	0	0.00%
	100	0	0.00%	50	100.00%	0	0.00%	23	46.00%	27	54.00%	0	0.00%
	200	4	8.00%	44	88.00%	2	4.00%	39	78.00%	10	20.00%	1	2.00%
	Sub total	4	1.60%	244	97.60%	2	0.80%	92	36.80%	157	62.80%	1	0.40%
IV	500	36	72.00%	8	16.00%	6	12.00%	48	96.00%	2	4.00%	0	0.00%
	1000	50	100.00%	0	0.00%	0	0.00%	50	100.00%	0	0.00%	0	0.00%
	Sub total	86	86.00%	8	8.00%	6	6.00%	98	98.00%	2	2.00%	0	0.00%
Total Average		163	23.29%	507	72.43%	30	4.29%	342	48.86%	355	50.71%	3	0.43%

follows:

$$\mathrm{ARD} = \frac{\left[\sum_{i=1}^{n} \frac{obj_i^h - obj_i^{best}}{obj_i^{best}}\right]}{n} \times 10,000‱$$

Here, obj_i^h is the value of the objective function in the i^{th} benchmark problem that was obtained by algorithm h; obj_i^{best} is the best value of the objective function in the i^{th} benchmark problem that was obtained by any of the compared algorithms, n is the number of benchmark problems under consideration, and ‱ is a per ten thousand sign.

B. PARAMETER VALUE DETERMINATION

Since all of the relevant parameters may influence the performance of the proposed BSA algorithm, extensive computational testing was carried out to evaluate them. In the preliminary tests, the following combinations of parameter values were used in 16 benchmark problems that were randomly selected from the four sets thereof, and each problem was solved by three independent applications of the proposed BSA algorithm: $I_{iter} \in \{500, 1000, 1500, 2000\}$; $B_{non-improving} \in \{5, 10, 15, 20\}$; $\eta_{\min} \in \{10, 20, 30\}$; $T_0 \in \{3, 5, 10, 15, 20\}$; $\alpha \in \{0.96, 0.97, 0.98, 0.99\}$, and $T_F \in \{0.01, 0.05, 0.10, 0.15, 0.20\}$. The test results showed that the best performance of the BSA algorithm was achieved within a reasonable computation time using $T_0 = 10, T_F = 0.1, I_{iter} = 1000$, $B_{non-improving} = 5$, $\eta_{\min} = 10$, and $\alpha = 0.98$. Accordingly, these parameter values were used in the experiments.

C. COMPUTATIONAL RESULTS

Table 3 presents the ARD and average running time (in seconds) to solve a problem of each size. The total average ARD for all 700 instances that was obtained using the proposed BSA algorithm is 0.007‱, whereas the corresponding values that were obtained by SA and ES are 0.809‱ and 9.921‱, respectively. Obviously, the proposed BSA algorithm outperforms the state-of-the-art ES algorithm and the traditional SA heuristic in solving the $1|RCDW|\sum(\alpha_j E_j + \beta_j T_j)$ problem. The computational times of the BSA and SA algorithms are almost equal because they apply the same termination condition. In contrast, the computational times of the BSA and SA algorithms are much shorter than that of ES when $N \geq 200$, indicating that the encoding scheme of BSA and SA is more efficient than that of ES. As shown in Fig. 4, compared with ES, the proposed BSA can provide smaller ARDs for all different job numbers. Fig. 4 shows that SA and BSA can provide almost the same ARDs when the number of jobs is smaller than 500.

To verify the effectiveness of the proposed BSA algorithm, paired t-tests are performed on the ARD obtained using this algorithm to compare it with those of ES and SA algorithms. Table 4 reveals that the proposed BSA algorithm significantly outperforms the SA and ES algorithms for $N = 500$ and 1000 with problem sets II and IV at the 95% confidence level.

TABLE 6. The best known solutions of benchmark problem sets I and II.

		Problem data set I					Problem set II	
K	(h_E, h_T)	$N=10$	$N=20$	$N=50$	$N=100$	$N=200$	$N=500$	$N=1000$
1	(0.1, 0.2)	**1896**	**4089**	**39461**	**139568***	**474405**	**2812324**	**13514502**
	(0.1, 0.3)	**1330**	**2713**	**28225**	**95211**	**324463**	**1915938**	**9315188**
	(0.2, 0.5)	**540**	**1162**	**12754**	**39487**	**136751**	**810403**	**3914623**
	(0.3, 0.4)	**919**	**2294**	**21110**	**72110**	**246659**	**1476558**	**6948451**
	(0.3, 0.5)	**587**	**1559**	**13971**	**45826**	158116	**944399**	**4404641**
2	(0.1, 0.2)	**947**	**8251**	**29043**	**120484**	**517302**	**3229169**	11733558
	(0.1, 0.3)	**539**	**5950**	**20133**	**82026**	**353001**	**2212062**	**8024367**
	(0.2, 0.5)	**191**	**2770**	**8468**	35290	**145421**	935690	**3368177**
	(0.3, 0.4)	**432**	**4482**	**15150**	**62421**	**267687**	**1679279**	**6098871**
	(0.3, 0.5)	**265**	**2923**	**9428**	**39705**	**169542**	**1076044**	**3882473**
3	(0.1, 0.2)	**1488**	**5881**	**33180**	**124317**	**466667**	**2950841**	11383782
	(0.1, 0.3)	**1012**	**4067**	**23020**	**86236**	**319357**	**2000406**	**7695608**
	(0.2, 0.5)	**398**	**1675**	**9962**	**38174**	134610	**834686**	**3140985**
	(0.3, 0.4)	**760**	**3035**	**17508**	**67046**	**244991**	**1534635**	**5778518**
	(0.3, 0.5)	**462**	**1998**	**11388**	**44048**	**156984**	**980797**	**3633073**
4	(0.1, 0.2)	**2128**	**8977**	**25856**	**122901**	**564830**	**3075510**	11201090
	(0.1, 0.3)	**1576**	**6609**	**17544**	**84112**	**396584**	**2075252**	**7590369**
	(0.2, 0.5)	**712**	**3113**	**7373**	**35498**	**176837**	**841660**	**3148904**
	(0.3, 0.4)	**1162**	**4830**	**13609**	**65057**	**304632**	**1567957**	**5800727**
	(0.3, 0.5)	**740**	**3210**	**8418**	**41584**	**200659**	**988910**	**3672325**
5	(0.1, 0.2)	**1150**	**4028**	**31456**	**119115**	**488829**	**2983984**	11857765
	(0.1, 0.3)	**755**	**2850**	**21689**	**82398**	**333482**	**2048254**	**8023851**
	(0.2, 0.5)	**284**	**1192**	**8947**	**34860**	**139842**	**858055**	**3287539**
	(0.3, 0.4)	**542**	**2112**	**15747**	**60963**	**254510**	**1534573**	**6087728**
	(0.3, 0.5)	**339**	**1341**	**9956**	38781	**162120**	**971873**	**3830152**
6	(0.1, 0.2)	**1479**	**6306**	**33452**	**133545**	**458004**	2654972	**11067696**
	(0.1, 0.3)	**1023**	**4247**	**23261**	**89563**	311824	1775879	**7484909**
	(0.2, 0.5)	**439**	**1557**	**10221**	**35146**	**126398**	**708038**	**3097750**
	(0.3, 0.4)	**779**	**3042**	**17392**	**65472**	**233366**	1328187	**5707324**
	(0.3, 0.5)	**500**	**1778**	**11178**	**40858**	**146243**	**827793**	**3627473**
7	(0.1, 0.2)	**2093**	**10204**	**42234**	**129849**	**428095**	**3027551**	**12684832**
	(0.1, 0.3)	**1521**	**7492**	**29274**	**90963**	**285328**	**2069491**	**8686112**
	(0.2, 0.5)	**717**	**3573**	**12000**	**39336**	**115477**	873675	**3651121**
	(0.3, 0.4)	**1190**	**5722**	**20696**	**67795**	**220289**	**1582907**	**6592156**
	(0.3, 0.5)	**809**	**3846**	**12935**	**43737**	139043	**1014404**	**4205935**
8	(0.1, 0.2)	**1644**	**3742**	**42218**	**153965**	**474446**	2976788	11686538
	(0.1, 0.3)	**1287**	**2519**	**28403**	**106517**	**320676**	1998263	**7940720**
	(0.2, 0.5)	**670**	**990**	**11154**	**44963**	**129912**	815282	**3279303**
	(0.3, 0.4)	**952**	**1801**	**20965**	**80269**	**235477**	**1506415**	6031322
	(0.3, 0.5)	**680**	**1069**	**12913**	**51414**	146955	**947726**	**3794560**
9	(0.1, 0.2)	**1466**	**3317**	**33222**	**111474**	**508275**	**3224615**	**11144396**
	(0.1, 0.3)	**1121**	**2342**	**23840**	**75152**	**350341**	**2205768**	**7539465**
	(0.2, 0.5)	**492**	**1056**	**10968**	31266	**147166**	**922679**	**3136842**
	(0.3, 0.4)	**772**	**1767**	**17972**	**57102**	**263243**	**1658772**	**5795388**
	(0.3, 0.5)	**513**	**1187**	**11935**	**36319**	**168041**	**1059669**	**3689098**
10	(0.1, 0.2)	**1835**	**4673**	**31492**	**112799**	**514927**	**2977821**	**11822252**
	(0.1, 0.3)	**1384**	**3266**	**22036**	**78670**	**358435**	**2031659**	**8028517**
	(0.2, 0.5)	**691**	**1355**	**9653**	**34068**	**157881**	**848389**	3293589
	(0.3, 0.4)	**1047**	**2419**	**16510**	**59922**	**273834**	**1539930**	6052594
	(0.3, 0.5)	**717**	**1474**	**10597**	**38639**	**178442**	**974261**	**3813645**

* Bold values indicate the best known solutions found by the BSA algorithm.

TABLE 7. The best known solutions of benchmark problem sets III and IV.

		Problem set III					Problem set IV	
K	(h_E, h_T)	$N=10$	$N=20$	$N=50$	$N=100$	$N=200$	$N=500$	$N=1000$
1	(0.4, 0.5)	**693**	**2115**	**16159**	**56462**	**195486**	**1188996**	**5298116**
	(0.4, 0.6)	**470**	**1478**	**10660**	**37147**	**129847**	**794983**	**3390555**
	(0.5, 0.6)	**632**	**2115**	**13458**	**51549**	**181331**	**1120240**	**4582742**
	(0.5, 0.7)	**462**	**1507**	**9492**	**36399**	127891	**786787**	**3175075**
	(0.6, 0.7)	**613**	**2148**	**13103**	**51398**	**181331**	**1120976**	**4539980**
2	(0.4, 0.5)	**408**	**3202**	**11321**	**47469**	**210182**	**1326189**	**4792029**
	(0.4, 0.6)	**265**	**2054**	**7240**	**31478**	**136800**	**878713**	**3125881**
	(0.5, 0.6)	**408**	**2529**	**9958**	**42562**	**189590**	**1227048**	**4340121**
	(0.5, 0.7)	**265**	**1675**	**6613**	**30468**	**132771**	**864513**	**3040446**
	(0.6, 0.7)	**408**	**2219**	**9670**	**42544**	**189489**	**1226290**	**4339326**
3	(0.4, 0.5)	**597**	**2590**	**13685**	**54382**	197114	**1239010**	**4547306**
	(0.4, 0.6)	**402**	**1783**	**8862**	**36553**	**131818**	**822586**	**2966739**
	(0.5, 0.6)	**554**	**2539**	**12000**	**49716**	**182261**	1165025	**4180534**
	(0.5, 0.7)	**402**	**1753**	**8379**	**35478**	**128261**	**815161**	**2901968**
	(0.6, 0.7)	**571**	**2501**	**11964**	**49716**	**181828**	**1165026**	**4179117**
4	(0.4, 0.5)	**808**	**3406**	**10651**	**52257**	**242258**	**1241531**	**4611941**
	(0.4, 0.6)	**545**	**2184**	**6842**	**34891**	**163792**	**821906**	**3042188**
	(0.5, 0.6)	**664**	**2585**	**9707**	**48432**	**219420**	**1165534**	**4292135**
	(0.5, 0.7)	**474**	**1664**	**6683**	**33703**	**158107**	816389	**3001860**
	(0.6, 0.7)	**620**	**2207**	**9630**	**48078**	**218734**	**1165534**	**4292137**
5	(0.4, 0.5)	**414**	**1660**	**11820**	**46227**	**201387**	**1177569**	**4810733**
	(0.4, 0.6)	**283**	**1124**	**7721**	**29439**	**132217**	**756670**	**3136870**
	(0.5, 0.6)	**377**	**1570**	**10525**	**39576**	**184939**	1037437	**4450742**
	(0.5, 0.7)	**269**	**1078**	**7463**	**26525**	**129898**	722465	**3086022**
	(0.6, 0.7)	**373**	**1542**	**10502**	**38437**	**184939**	**1035553**	**4452216**
6	(0.4, 0.5)	**600**	**2227**	**13004**	**50347**	**181802**	**1046776**	**4577050**
	(0.4, 0.6)	**417**	**1418**	**8121**	**32258**	116811	**691409**	**3045836**
	(0.5, 0.6)	**575**	**2075**	**10342**	**44289**	**165372**	**988565**	**4297410**
	(0.5, 0.7)	**397**	**1430**	**6696**	**30557**	**114213**	**687317**	**3011451**
	(0.6, 0.7)	**545**	**2114**	**9545**	**43879**	**165372**	989247	**4296003**
7	(0.4, 0.5)	**983**	**4228**	**15039**	**51898**	**180276**	**1259415**	**5184062**
	(0.4, 0.6)	**688**	**2755**	**9586**	**34124**	**119709**	833425	**3392192**
	(0.5, 0.6)	**866**	**3307**	**12799**	**45306**	**172017**	**1162321**	**4675651**
	(0.5, 0.7)	**609**	**2189**	**8895**	**31823**	**119341**	**817557**	**3272245**
	(0.6, 0.7)	**812**	**2885**	**12585**	**44608**	**172017**	**1161582**	**4663784**
8	(0.4, 0.5)	**735**	**1244**	**16283**	**62674**	**179248**	**1181993**	**4705887**
	(0.4, 0.6)	**448**	**780**	**10701**	**41454**	**113806**	**768471**	**3045284**
	(0.5, 0.6)	**494**	**1102**	**15228**	**58020**	**157327**	**1080104**	**4276411**
	(0.5, 0.7)	**318**	**739**	**10582**	**41074**	**108709**	**752411**	**2969460**
	(0.6, 0.7)	**421**	**1096**	**15145**	**58184**	**156738**	**1080817**	**4274879**
9	(0.4, 0.5)	**571**	**1465**	**13592**	**45447**	**205130**	**1304375**	**4673809**
	(0.4, 0.6)	**335**	**1041**	**8964**	**29422**	**133219**	**861164**	**3093889**
	(0.5, 0.6)	**413**	**1428**	**11129**	**41001**	**182230**	**1198181**	**4366051**
	(0.5, 0.7)	**278**	**1038**	**7443**	**28315**	**127845**	**842986**	**3049762**
	(0.6, 0.7)	**394**	**1428**	**10203**	**40962**	**181889**	**1197429**	**4366051**
10	(0.4, 0.5)	**763**	**1804**	**12409**	**47525**	**216838**	**1201709**	**4738093**
	(0.4, 0.6)	**507**	**1095**	**7800**	**31158**	**143958**	**776891**	**3074400**
	(0.5, 0.6)	**584**	**1499**	**10176**	**43228**	**193926**	**1072017**	**4322155**
	(0.5, 0.7)	**387**	**872**	**7024**	**30141**	**136437**	**747205**	**3012394**
	(0.6,0.7)	**507**	**1289**	**10090**	**43086**	**192192**	**1070428**	**4322154**

* Bold values indicate the best known solutions found by the BSA algorithm.

However, the BSA algorithm is not statistically better than the SA algorithm for $N = 10$, 20, 50, 100 and 200 with problem sets I and III, perhaps because the SA algorithm also performs well when it is applied to small and medium-sized problems. Nevertheless, the BAS is indeed statistically better than ES, even when N is 50.

The computational results are analyzed with a focus on the number of solutions obtained by the proposed BSA algorithm; they are better, equal to, or worse than those obtained by the SA and EA algorithms. As shown in Table 5, the proposed BSA algorithm is better than, equal to, and worse than the SA algorithm in 163 out of 700, 507 out of 700, and 30 out of 700 benchmark problems, respectively. The proposed BSA algorithm is better than, equal to, and worse than the ES algorithm in 342 out of 700, 355 out of 700, and 3 out of 770 benchmark problems, respectively. In the benchmark problems with $N = 500$ and 1000, 67% and 90%, respectively, the solutions obtained using the proposed BSA algorithm are better than those obtained using the SA algorithm. In the test instances with $N = 200$, 500 and 1000, 67%, 98% and 100%, respectively, of the solutions obtained using the proposed BSA algorithm are better than those obtained using the EA algorithm. The analytical results reveal that BSA outperforms SA in most benchmark problems with $N \geq 500$, whereas BSA outperforms ES in most of benchmark problems with $N \geq 200$. As shown in Fig. 5, compared with ES, the proposed BSA can provide much better solutions when the number of jobs increases. Fig. 5 shows that compared with SA, the BSA can provide better solutions when the number of jobs is equal to and larger than 200.

To provide a benchmark for future research, Appendix Table 6 presents the best known solutions of the 250 and 100 benchmark problems in problem sets I and II, respectively, while Appendix Table 7 presents the best known solutions of the 250 and 100 benchmark problems in problems sets III and IV, respectively.

VI. CONCLUSION AND RECOMMENDATIONS FOR FUTURE RESEARCH

This paper concerns the $1|RCDW|\sum(\alpha_j E_j + \beta_j T_j)$ problem, which is not only theoretically but also practically interesting. We present a complete perspective on all possible optimal sequences associated with various straddling jobs and production waiting times. An effective and efficient BSA algorithm, which includes a backtracking mechanism and an effective coding scheme, is proposed to solve the above problem. Computational experiments that involve extensive benchmark test instances demonstrate that the proposed backtracking mechanism can improve the performance of the SA algorithm and make the proposed BSA algorithm significantly outperform the best available algorithm published in the literature. This research contributes by providing useful optimization approaches to the $1|RCDW|\sum(\alpha_j E_j + \beta_j T_j)$ problem. Since few algorithms are currently available for solving this strongly $\mathcal{NP}$-complete problem, the presented approaches can help practitioners solve real-world $1|RCDW|\sum(\alpha_j E_j + \beta_j T_j)$ problems with respect to the JIT manufacturing system.

Many interesting related topics warrant further investigation. First, the problem herein should be extended to include various plausible objectives. Second, more effective and efficient meta-heuristics for solving the $1|RCDW|\sum(\alpha_j E_j + \beta_j T_j)$ problem warrant further exploration. Third, more research is needed to develop exact methods for solving the $1|RCDW|\sum(\alpha_j E_j + \beta_j T_j)$ problem. Fourth, further investigations of problem variants with additional realistic constraints, such as sequence-dependent setup times and release times, would support a rich body of future studies. Fifth, the SMSP with an RCDW in which a bi-objective function value is minimized, would be an interesting target of research. Finally, future research could consider other production systems (such as flow-shop and job-shop) that involve an RCDW.

APPENDIX

See Tables 6 and 7.

REFERENCES

[1] Y. Chen, L.-H. Su, Y.-C. Tsai, S. Huang, and F.-D. Chou, "Scheduling jobs on a single machine with dirt cleaning consideration to minimize total completion time," *IEEE Access*, vol. 7, pp. 22290–22300, 2019.

[2] J. H. Blackstone, D. T. Phillips, and G. L. Hogg, "A state-of-the-art survey of dispatching rules for manufacturing job shop operations," *Int. J. Prod. Res.*, vol. 20, no. 1, pp. 27–45, 1982.

[3] S. S. Panwalkar and W. Iskander, "A survey of scheduling rules," *Oper. Res.*, vol. 25, no. 1, pp. 45–61, Jan./Feb. 1977.

[4] S.-W. Lin and K.-C. Ying, "A hybrid approach for single-machine tardiness problems with sequence-dependent setup times," *J. Oper. Res. Soc.*, vol. 59, no. 8, pp. 1109–1119, 2008.

[5] C.-C. Lu, S.-W. Lin, and K.-C. Ying, "Robust single machine scheduling for minimizing total flow time in the presence of uncertain processing times," *Comput. Ind. Eng.*, vol. 74, pp. 102–110, Aug. 2014.

[6] C. Gahma, J. J. Kanet, and A. Tuma, "On the flexibility of a decision theory-based heuristic for single machine scheduling," *Comput. Oper. Res.*, vol. 101, pp. 103–115, Jan. 2019.

[7] N. C. O. da Silva, C. T. Scarpin, J. E. Pécora, Jr., and A. Ruiz, "Online single machine scheduling with setup times depending on the jobs sequence," *Comput. Ind. Eng.*, vol. 129, pp. 251–258, Mar. 2019.

[8] K. R. Baker and G. D. Scudder, "Sequencing with earliness and tardiness penalties: A review," *Oper. Res.*, vol. 38, no. 1, pp. 22–36, Feb. 1990.

[9] S.-W. Lin, S.-Y. Chou, and K.-C. Ying, "A sequential exchange approach for minimizing earliness–tardiness penalties of single-machine scheduling with a common due date," *Eur. J. Oper. Res.*, vol. 177, no. 2, pp. 1294–1301, Mar. 2007.

[10] J. B. Sidney, "Optimal single-machine scheduling with earliness and tardiness penalties," *Oper. Res.*, vol. 25, no. 1, pp. 62–69, Jan./Feb. 1977.

[11] G. V. Gens and E. V. Levner, "Fast approximation algorithm for job sequencing with deadlines," *Discrete Appl. Math.*, vol. 3, no. 4, pp. 313–318, Nov. 1981.

[12] J. J. Kanet, "Minimizing the average deviation of job completion times about a common due date," *Nav. Res. Logistics Quart.*, vol. 28, no. 4, pp. 643–651, Dec. 1981.

[13] M. Raghavachari, "A V-shape property of optimal schedule of jobs about a common due date," *Eur. J. Oper. Res.*, vol. 23, no. 3, pp. 401–402, 1986.

[14] A. M. Krieger and M. Raghavachari, "V-shape property for optimal schedules with monotone penalty functions," *Comput. Oper. Res.*, vol. 19, no. 6, pp. 533–534, Aug. 1992.

[15] E. Gerstl and G. Mosheiov, "Single machine just-in-time scheduling problems with two competing agents," *Nav. Res. Logistics*, vol. 61, no. 1, pp. 1–16, Feb. 2014.

[16] E. Gerstl and G. Mosheiov, "Scheduling with a due-window for acceptable lead-times," *J. Oper. Res. Soc.*, vol. 66, no. 9, pp. 1578–1588, 2015.

[17] Y.-Y. Lu, J.-J. Wang, and X. Huang, "Scheduling jobs with position and sum-of-processing-time based processing times," *Appl. Math. Model.*, vol. 39, no. 14, pp. 4013–4021, Jul. 2015.

[18] N. G. Hall, W. Kubiak, and S. P. Sethi, "Earliness–tardiness scheduling problems, II: Deviation of completion times about a restrictive common due date," *Oper. Res.*, vol. 39, no. 5, pp. 847–856, 1991.

[19] N. G. Hall and M. E. Posner, "Earliness-tardiness scheduling problems, I: Weighted deviation of completion times about a common due date," *Oper. Res.*, vol. 39, no. 5, pp. 836–846, 1991.

[20] V. Gordon, J.-M. Proth, and C. Chu, "A survey of the state-of-the-art of common due date assignment and scheduling research," *Eur. J. Oper. Res.*, vol. 139, no. 1, pp. 1–25, May 2002.

[21] G. Moslehi, M. Mahnam, M. Amin-Nayeri, and A. Azaron, "A branch-and-bound algorithm to minimise the sum of maximum earliness and tardiness in the single machine," *Int. J. Oper. Res.*, vol. 8, no. 4, pp. 458–482, Jan. 2010.

[22] Z.-J. Lee, C.-C. Chuang, and K.-C. Ying, "An intelligent algorithm for scheduling jobs on a single machine with a common due date," in *Knowledge-Based Intelligent Information and Engineering Systems*. Cham, Switzerland: Springer, 2007, pp. 689–695.

[23] C. Low, R.-K. Li, G.-H. Wu, and C.-L. Huang, "Minimizing the sum of absolute deviations under a common due date for a single-machine scheduling problem with availability constraints," *J. Ind. Prod. Eng.*, vol. 32, no. 3, pp. 204–217, 2015.

[24] Y. Yin, T. C. E. Cheng, C.-C. Wu, and S.-R. Cheng, "Single-machine batch delivery scheduling and common due-date assignment with a rate-modifying activity," *Int. J. Prod. Res.*, vol. 52, no. 19, pp. 5583–5596, Feb. 2014.

[25] A. Janiak, W. A. Janiak, T. Krysiak, and T. Kwiatkowski, "A survey on scheduling problems with due windows," *Eur. J. Oper. Res.*, vol. 242, no. 2, pp. 347–357, Apr. 2015.

[26] B. Mor, "Single-machine minmax common due-window assignment and scheduling problems with convex resource allocation," *Eng. Optim.*, vol. 51, no. 7, pp. 1251–1267, 2019.

[27] D. Biskup and M. Feldmann, "On scheduling around large restrictive common due windows," *Eur. J. Oper. Res.*, vol. 162, no. 3, pp. 740–761, May 2005.

[28] F. D. Anger, C. Y. Lee, and L. A. Martin-Vega, "Single machine scheduling with tight windows," Dept. Ind. Syst. Eng., Univ. Florida, Univ. Gainesville, FL, USA, Tech. Rep. 16–86, 1986.

[29] F.-J. Krömer and C.-Y. Lee, "Common due-window scheduling," *Prod. Oper. Manag.*, vol. 2, no. 4, pp. 262–275, Dec. 1993.

[30] S. D. Liman and S. Ramaswamy, "Earliness-tardiness scheduling problems with a common delivery window," *Oper. Res. Lett.*, vol. 15, no. 4, pp. 195–203, May 1994.

[31] J. A. Ventura and M. X. Weng, "Single machine scheduling with a common delivery window," *J. Oper. Res. Soc.*, vol. 47, no. 3, pp. 424–434, Mar. 1996.

[32] W.-S. Yoo and L. A. Martin-Vega, "Scheduling single-machine problems for on-time delivery," *Comput. Ind. Eng.*, vol. 39, nos. 3–4, pp. 371–392, Apr. 2001.

[33] J. M. Moore, "An n job, one machine sequencing algorithm for minimizing the number of late jobs," *Manage. Sci.*, vol. 15, no. 1, pp. 102–109, Sep. 1968.

[34] W. K. Yeung, C. Oğuz, and T. C. E. Cheng, "Single-machine scheduling with a common due window," *Comput. Oper. Res.*, vol. 28, no. 2, pp. 157–175, Feb. 2001.

[35] M. Azizoglu and S. Webster, "Scheduling about an unrestricted common due window with arbitrary earliness/tardiness penalty rates," *IIE Trans.*, vol. 29, no. 11, pp. 1001–1006, 1997.

[36] C.-L. Li, "Improved algorithms for single-machine common due window assignment and scheduling with batch deliveries," *Theor. Comput. Sci.*, vol. 570, pp. 30–39, Mar. 2015.

[37] L. Liu, J.-J. Wang, and X.-Y. Wang, "Single machine due-window assignment scheduling with resource-dependent processing times to minimise total resource consumption cost," *Int. J. Prod. Res.*, vol. 54, no. 4, pp. 1186–1195, 2016.

[38] C. Zhao, C.-J. Hsu, W.-C. Lin, W.-H. Wu, and C.-C. Wu, "An investigation of single-machine due-window assignment with time-dependent processing times and a controllable rate-modifying activity," *Comput. J.*, vol. 60, no. 9, pp. 1353–1362, Feb. 2017.

[39] L. Liu, J.-J. Wang, F. Liu, and M. Liu, "Single machine due window assignment and resource allocation scheduling problems with learning and general positional effects," *J. Manuf. Syst.*, vol. 43, no. 1, pp. 1–14, Apr. 2017.

[40] X. Zhang, W.-C. Lin, W.-H. Wu, and C.-C. Wu, "Single-machine common/slack due window assignment problems with linear decreasing processing times," *Eng. Optim.*, vol. 49, no. 8, pp. 1388–1400, 2017.

[41] B. Mor, "Minmax common due-window assignment and scheduling on a single machine with two competing agents," *J. Oper. Res. Soc.*, vol. 69, no. 4, pp. 589–602, 2018.

[42] N. Yin, "Single machine due window assignment resource allocation scheduling with job-dependent learning effect," *J. Appl. Math. Comput.*, vol. 56, no. 13, pp. 715–725, Feb. 2018.

[43] D. Wang and Z. Li, "Bicriterion scheduling with a negotiable common due window and resource-dependent processing times," *Inf. Sci.*, vol. 478, pp. 258–274, Apr. 2019.

[44] J.-B. Wang, B. Zhang, L. Li, D. Bai, and Y.-B. Feng, "Due-window assignment scheduling problems with position-dependent weights on a single machine," *Eng. Optim.*, to be published. doi: 10.1080/0305215x.2019.1577411.

[45] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey," *Ann. Math.*, vol. 5, pp. 287–326, 1979.

[46] M. Feldmann and D. Biskup, "Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches," *Comput. Ind. Eng.*, vol. 44, no. 2, pp. 307–323, Feb. 2003.

[47] G. Jianlan, C. Yuqiang, and H. Xuanzi, "Implementation and improvement of simulated annealing algorithm in neural net," in *Proc. Int. Conf. Comput. Intell. Secur.*, Dec. 2010, pp. 519–522.

[48] A. Askarzadeh, L. D. S. Coelho, C. E. Klein, and V. C. Mariani, "A population-based simulated annealing algorithm for global optimization," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2016, pp. 4626–4633.

[49] L. Qin, J. Wang, H. Li, Y. Sun, and S. Li, "An approach to improve the performance of simulated annealing algorithm utilizing the variable universe adaptive fuzzy logic system," *IEEE Access*, vol. 5, pp. 18155–18165, 2017.

[50] S. H. Zhan, Z. J. Zhang, L. J. Wang, and Y. W. Zhong, "List-based simulated annealing algorithm with hybrid greedy repair and optimization operator for 0-1 Knapsack problem," *IEEE Access*, vol. 6, pp. 54447–54458, 2018.

[51] A. Franzin and T. Stützle, "Revisiting simulated annealing: A component-based analysis," *Comput. Oper. Res.*, vol. 104, pp. 191–206, Apr. 2019.

[52] S.-W. Lin, K.-C. Ying, Y.-I. Chiang, and W.-J. Wu, "Minimising total weighted earliness and tardiness penalties on identical parallel machines using a fast ruin-and-recreate algorithm," *Int. J. Prod. Res.*, vol. 54, no. 22, pp. 6879–6890, 2016.

[53] D. Biskup and M. Feldmann, "Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates," *Comput. Oper. Res.*, vol. 28, no. 8, pp. 787–801, Jul. 2001.

[54] K.-C. Ying, S.-W. Lin, and C.-C. Lu, "Effective dynamic dispatching rule and constructive heuristic for solving single-machine scheduling problems with a common due window," *Int. J. Oper. Res.*, vol. 55, no. 6, pp. 1707–1719, 2017.

SHIH WEI LIN received the bachelor's, master's, and Ph.D. degrees in industrial management from the National Taiwan University of Science and Technology, Taiwan, in 1996, 1998, and 2000, respectively. He is currently a Professor with the Department of Information Management, Chang Gung University, Taiwan. He is also with Department of Neurology, Linkou Chang Gung Memorial Hospital, Taoyuan, Taiwan, and the Department of Industrial Engineering and Management, Ming Chi University of Technology, Taipei, Taiwan. His articles have appeared in *Computers and Operations Research*, the *European Journal of Operational Research*, the *Journal of the Operational Research Society*, the *European Journal of Industrial Engineering*, the *International Journal of Production Research*, the *International Journal of Advanced Manufacturing Technology*, *Knowledge and Information Systems*, and *Applied Soft Computing*, *Applied Intelligence*, and *Expert Systems with Applications*. His current research interests include meta-heuristics and data mining.

KUO-CHING YING is currently a Distinguished Professor with the Department of Industrial Engineering and Management, National Taipei University of Technology, Taiwan. He has published over 100 academic research articles in refereed international journals, such as *Applied Intelligence*, *Applied Soft Computing*, *Computers and Operations Research*, *Computers and Industrial Engineering*, the *European Journal of Industrial Engineering*, the *European Journal of Operational Research*, IEEE ACCESS, the *International Journal of Production Economics*, the *International Journal of Advanced Manufacturing Technology*, *International Journal of Innovational Computing*, *Information and Control*, the *International Journal of Production Research*, the *Journal of the Operational Research Society*, *OMEGA—The International Journal of Management Sciences, Production Planning and Control*, and *Transportation Research Part E: Logistics and Transport Review*. His research interests include operations scheduling and combinatorial optimization. He is on the editorial board of ten prestigious journals and a Reviewer of 45 journals.

WEN-JIE WU received the B.S. degree in computer science and engineering from Tatung University, Taipei, Taiwan, in 1996, and the M.S. and Ph.D. degrees in computer science and information engineering from National Chung Cheng University, Chiayi, Taiwan, in 1998 and 2003, respectively. He is currently an Assistant Professor with the Department of Information Management, Chang Gung University, Taiwan. His research interests include image processing, medical computer-aided diagnosis systems, and data mining.

CHEN-YANG CHENG received the Ph.D. degree in industrial and manufacturing engineering from Penn State University. He is currently a Professor with the Department of Industrial Engineering and Management, National Taipei University of Technology. His research interests include computer integrated manufacturing, human–computer interaction, distributed systems and control, and intelligent systems.